

Application for United States Letters Patent

for

**MEMORY MANAGEMENT SYSTEM AND METHOD  
PROVIDING LINEAR ADDRESS BASED  
MEMORY ACCESS SECURITY**

by

**Brian C. Barnes**

**Geoffrey S. Strongin**

**Rodney W. Schmidt**

EXPRESS MAIL MAILING LABEL

NUMBER EL798365109US

DATE OF DEPOSIT 13 November 2001

I hereby certify that this paper or fee is being deposited with the United States Postal Service "EXPRESS MAIL POST OFFICE TO ADDRESSEE" service under 37 C.F.R. 1.10 on the date indicated above and is addressed to Assistant Commissioner for Patents, Washington D.C. 20231.

*Sharon Darte*

SIGNATURE

**MEMORY MANAGEMENT SYSTEM AND METHOD  
PROVIDING LINEAR ADDRESS BASED  
MEMORY ACCESS SECURITY**

**BACKGROUND OF THE INVENTION**

**1. REFERENCES**

This patent application is related to a co-pending patent application serial no. \_\_\_\_\_ (Attorney Reference Number 2000.056600/TT4086) entitled "Memory Management System And Method Providing Physical Address Based Memory Access Security" by Brian C. Barnes, Geoffrey S. Strongin, and Rodney W. Schmidt, filed on the same day as the present patent application.

**2. FIELD OF THE INVENTION**

This invention relates generally to memory management systems and methods, and, more particularly, to memory management systems and methods that provide protection for data stored within a memory.

**3. DESCRIPTION OF THE RELATED ART**

A typical computer system includes a memory hierarchy to obtain a relatively high level of performance at a relatively low cost. Instructions of several different software programs are typically stored on a relatively large but slow non-volatile storage unit (e.g., a disk drive unit). When a user selects one of the programs for execution, the instructions of the selected program are copied into a main memory unit (e.g., random access memory (RAM)), and a central processing unit (CPU) obtains the instructions of the selected program from the main memory unit. A well-known virtual memory management technique allows the CPU to access data structures larger in size than that of the main memory unit by storing only a portion of the data structures within the main memory unit at any given time.

Remainders of the data structures are stored within the relatively large but slow non-volatile storage unit, and are copied into the main memory unit only when needed.

Virtual memory is typically implemented by dividing an address space of the CPU into multiple blocks called page frames or “pages.” Only data corresponding to a portion of the pages is stored within the main memory unit at any given time. When the CPU generates an address within a given page, and a copy of that page is not located within the main memory unit, the required page of data is copied from the relatively large but slow non-volatile storage unit into the main memory unit. In the process, another page of data may be copied from the main memory unit to the non-volatile storage unit to make room for the required page.

The popular 80x86 (x86) processor architecture includes specialized hardware elements to support a protected virtual address mode (i.e., a protected mode). Figs. 1-3 will now be used to describe how an x86 processor implements both virtual memory and memory protection features. Fig. 1 is a diagram of a well-known linear-to-physical address translation mechanism 100 of the x86 processor architecture. An address translation mechanism 100 is embodied within an x86 processor, and involves a linear address 102 produced within the x86 processor, a page table directory (i.e., a page directory) 104, multiple page tables including a page table 106, multiple page frames including a page frame 108, and a control register (CR3) 110. The page directory 104 and the multiple page tables are paged memory data structures created and maintained by operating system software (i.e., an operating system). The page directory 104 is always located within the memory (e.g., the main memory unit). For simplicity, the page table 106 and the page frame 108 will also be assumed to reside in the memory.

As indicated in Fig. 1, the linear address 102 is divided into three portions to accomplish the linear-to-physical address translation. The highest ordered bits of the CR3

110 are used to store a page directory base register. The page directory base register is a base address of a memory page containing the page directory 104. The page directory 104 includes multiple page directory entries, including a page directory entry 112. An upper “directory index” portion of the linear address 102, including the highest ordered or most significant bits of the linear address 102, is used as an index into the page directory 104. The page directory entry 112 is selected from within the page directory 104 using the page directory base address of the CR3 110 and the upper “directory index” portion of the linear address 102.

Fig. 2 is a diagram of a page directory entry format 200 of the x86 processor architecture. As indicated in Fig. 2, the highest ordered (i.e., most significant) bits of a given page directory entry contain a page table base address, where the page table base address is a base address of a memory page containing a corresponding page table. The page table base address of the page directory entry 112 is used to select the corresponding page table 106.

Referring back to Fig. 1, the page table 106 includes multiple page table entries, including a page table entry 114. A middle “table index” portion of the linear address 102 is used as an index into the page table 106, thereby selecting the page table entry 114. Fig. 3 is a diagram of a page table entry format 300 of the x86 processor architecture. As indicated in Fig. 3, the highest ordered (i.e., most significant) bits of a given page table entry contain a page frame base address, where the page frame base address is a base address of a corresponding page frame.

Referring again to Fig. 1, the page frame base address of the page table entry 114 is used to select the corresponding page frame 108. The page frame 108 includes multiple memory locations. A lower or “offset” portion of the linear address 102 is used as an index into the page frame 108. When combined, the page frame base address of the page table entry 114 and the offset portion of the linear address 102 produce the physical address

corresponding to the linear address 102, and indicate a memory location 116 within the page frame 108. The memory location 116 has the physical address resulting from the linear-to-physical address translation.

Regarding the memory protection features, the page directory entry format 200 of Fig. 2 and the page table entry format 300 of Fig. 3 include a user/supervisor (U/S) bit and a read/write (R/W) bit. The contents of the U/S and R/W bits are used by the operating system to protect corresponding page frames (i.e., memory pages) from unauthorized access. U/S=0 is used to denote operating system memory pages, and corresponds to a “supervisor” level of the operating system. The supervisor level of the operating system corresponds to a current privilege level 0 (CPL0) of software programs and routines executed by the x86 processor. U/S=1 is used to indicate user memory pages, and corresponds to a “user” level of the operating system. The user level of the operating system corresponds to CPL3 of the x86 processor. (The user level may also correspond to CPL1 and/or CPL2 of the x86 processor.)

The R/W bit is used to indicate types of accesses allowed to the corresponding memory page. R/W=0 indicates the only read accesses are allowed to the corresponding memory page (i.e., the corresponding memory page is “read-only”). R/W=1 indicates that both read and write accesses are allowed to the corresponding memory page (i.e., the corresponding memory page is “read-write”).

During the linear-to-physical address translation operation of Fig. 1, the contents of the U/S bits of the page directory entry 112 and the page table entry 114, corresponding to the page frame 108, are logically ANDed to determine if the access to the page frame 108 is authorized. Similarly, the contents of the R/W bits of the page directory entry 112 and the page table entry 114 are logically ANDed to determine if the access to the page frame 108 is authorized. If the logical combinations of the U/S and R/W bits indicate the access to the page frame 108 is authorized, the memory location 116 is accessed using the physical

address. On the other hand, if the logical combinations of the U/S and R/W bits indicate that the access to the page frame 108 is not authorized, the memory location 116 is not accessed, and a protection fault indication is signaled.

Unfortunately, the above described memory protection mechanisms of the x86 processor architecture are not sufficient to protect data stored in the memory. For example, any software program or routine executing at the supervisor level (e.g., having a CPL of 0) can access any portion of the memory, and can modify (i.e., write to) any portion of the memory that is not marked "read-only" (R/W=0). In addition, by virtue of executing at the supervisor level, the software program or routine can change the attributes (i.e., the U/S and R/W bits) of any portion of the memory. The software program or routine can thus change any portion of the memory marked "read-only" to "read-write" (R/W=1), and then proceed to modify that portion of the memory.

The present invention is directed to a method that may solve, or at least reduce, some or all of the aforementioned problems, and systems incorporating the method.

### **SUMMARY OF THE INVENTION**

A memory management unit is disclosed for managing a memory storing data arranged within a plurality of memory pages. The memory management unit includes a security check unit receiving a linear address generated during execution of a current instruction. The linear address has a corresponding physical address residing within a selected memory page. The security check unit uses the linear address to access one or more security attribute data structures located in the memory to obtain a security attribute of the selected memory page. The security check unit compares a numerical value conveyed by a security attribute of the current instruction to a numerical value conveyed by the security attribute of the selected memory page, and produces an output signal dependent upon a result of the comparison. The memory management unit accesses the selected memory page dependent upon the output signal.

A central processing unit (CPU) is described including an execution unit and the above described memory management unit (MMU). The execution unit fetches instructions from a memory and executes the instructions. The MMU manages the memory, and is configurable to manage the memory such that the memory stores data arranged within multiple memory pages. The MMU includes the above described security check unit. A computer system is described including the memory, the CPU, and the MMU.

A method is described for providing access security for a memory used to store data arranged within a plurality of memory pages. The method includes receiving a linear address produced during execution of an instruction, and a security attribute of the instruction, wherein the instruction resides in a first memory page. The linear address is used to access one or more paged memory data structures located in the memory to obtain a base address of a selected memory page, and security attributes of the selected memory page. If the security attribute of the instruction (e.g., a current privilege level or CPL of the instruction as defined by the x86 processor architecture) and the security attributes of the selected memory page (e.g., x86 U/S and R/W bits of the selected memory page) indicate the access is authorized, the base address of the selected memory page is combined with an offset to produce a physical address within the selected memory page. If the security attribute of the instruction and the security attributes of the selected memory page indicate the access is not authorized, a fault signal (e.g., an x86 GPF signal) is generated.

The linear address is also used to access one or more security attribute data structures located in the memory to obtain an additional security attribute of the first memory page, and an additional security attribute of the selected memory page. A numerical value conveyed by an additional security attribute of the first memory page is compared to a numerical value conveyed by the additional security attribute of selected memory page. The selected memory page is accessed dependent upon a result of the comparing of the numerical values conveyed

by the security attribute of the first memory page and the additional security attribute of selected memory page.

### **BRIEF DESCRIPTION OF THE DRAWINGS**

The invention may be understood by reference to the following description taken in conjunction with the accompanying drawings, in which like reference numerals identify similar elements, and in which:

Fig. 1 is a diagram of a well-known linear-to-physical address translation mechanism of the x86 processor architecture;

Fig. 2 is a diagram of a page directory entry format of the x86 processor architecture;

Fig. 3 is a diagram of a page table entry format of the x86 processor architecture;

Fig. 4 is a diagram of one embodiment of a computer system including a CPU, wherein the CPU includes a CPU security check unit (SCU);

Fig. 5 is a diagram illustrating relationships between various hardware and software components of the computer system of Fig. 4;

Fig. 6 is a diagram of one embodiment of the CPU of the computer system of Fig. 4, wherein the CPU includes a memory management unit (MMU);

Fig. 7 is a diagram of one embodiment of the MMU of Fig. 6, wherein the MMU includes a paging unit coupled to the CPU SCU;

Fig. 8 is a diagram of one embodiment of the CPU SCU of Fig. 7;

Fig. 9 is a diagram of one embodiment of a mechanism for accessing a security attribute table (SAT) entry of a given memory page to obtain additional security information of the given memory page;

Fig. 10 is a diagram of one embodiment of a SAT default register;

Fig. 11 is a diagram of one embodiment of a SAT directory entry format;

Fig. 12 is a diagram of one embodiment of a SAT entry format; and



Figs. 13A and 13B in combination form a flow chart of one embodiment of a method for managing a memory used to store data arranged within multiple memory pages.

While the invention is susceptible to various modifications and alternative forms, specific embodiments thereof have been shown by way of example in the drawings and are herein described in detail. It should be understood, however, that the description herein of specific embodiments is not intended to limit the invention to the particular forms disclosed, but on the contrary, the intention is to cover all modifications, equivalents, and alternatives falling within the spirit and scope of the invention as defined by the appended claims.

#### **DETAILED DESCRIPTION OF SPECIFIC EMBODIMENTS**

Illustrative embodiments of the invention are described below. In the interest of clarity, not all features of an actual implementation are described in this specification. It will, of course, be appreciated that in the development of any such actual embodiment, numerous implementation-specific decisions must be made to achieve the developers' specific goals, such as compliance with system-related and business-related constraints, which will vary from one implementation to another. Moreover, it will be appreciated that such a development effort might be complex and time-consuming, but would nevertheless be a routine undertaking for those of ordinary skill in the art having the benefit of this disclosure.

Fig. 4 is a diagram of one embodiment of a computer system 400 including a CPU 402, a system or "host" bridge 404, a memory 406, a first device bus 408 (e.g., a peripheral component interconnect or PCI bus), a device bus bridge 410, a second device bus 412 (e.g., an industry standard architecture or ISA bus), and four device hardware units 414A-414D. The host bridge 404 is coupled to the CPU 402, the memory 406, and the first device bus 408. The host bridge 404 translates signals between the CPU 402 and the first device bus 408, and operably couples the memory 406 to the CPU 402 and to the first device bus 408. The device bus bridge 410 is coupled between the first device bus 408 and the second device bus 412, and translates signals between the first device bus 408 and the second device bus

412. In the embodiment of Fig. 4, the device hardware units 414A and 414B are coupled to the first device bus 408, and the device hardware units 414C and 414D are coupled to the second device bus 412. One or more of the device hardware units 414A-414D may be, for example, storage devices (e.g., hard disk drives, floppy drives, and CD-ROM drives),  
5 communication devices (e.g., modems and network adapters), or input/output devices (e.g., video devices, audio devices, and printers).

In the embodiment of Fig. 4, the CPU 402 includes a CPU security check unit (SCU) 416. As will be described in detail below, the CPU SCU 416 protects the memory 406 from unauthorized accesses generated by the CPU 402. It is noted that in other embodiments, the host bridge 404 may be part of the CPU 402 as indicated in Fig. 4.

Fig. 5 is a diagram illustrating relationships between various hardware and software components of the computer system 400 of Fig. 4. In the embodiment of Fig. 5, multiple application programs 500, an operating system 502, a security kernel 504, and device drivers 506A-506D are stored in the memory 406. The application programs 500, the operating system 502, the security kernel 504, and the device drivers 506A-506D include instructions executed by the CPU 402. The operating system 502 provides a user interface and software “platform” on top of which the application programs 500 run. The operating system 502 may also provide, for example, basic support functions, including file system management, process management, and input/output (I/O) control.

The operating system 502 may also provide basic security functions. For example, the CPU 402 (Fig. 4) may be an x86 processor that executes instructions of the x86 instruction set. In this situation, the CPU 402 may include specialized hardware elements to provide both virtual memory and physical memory protection features in the protected mode as described above. The operating system 502 may be, for example, one of the Windows®  
25 family of operating systems (Microsoft Corp., Redmond, WA) that operates the CPU 402 in

the protected mode, and uses the specialized hardware elements of the CPU 402 to provide both virtual memory and memory protection in the protected mode.

As will be described in more detail below, the security kernel 504 provides additional security functions above the security functions provided by the operating system 502 to protect data stored in the memory 406 from unauthorized access. As indicated in Fig. 5, the security kernel 504 is coupled to the CPU SCU 416. As will be described in detail below, the CPU SCU 416 monitors all software-initiated accesses to the memory 406, and allows only authorized accesses to the memory 406.

In the embodiment of Fig. 5, the device drivers 506A-506D are operationally associated with, and coupled to, the respective corresponding device hardware units 414A-414D. The device hardware units 414A and 414D may be, for example, “secure” devices, and the corresponding device drivers 506A and 506D may be “secure” device drivers. The security kernel 504 is coupled between the operating system 502 and the secure device drivers 506A and 506D, and may monitor all accesses by the application programs 500 and the operating system 502 to secure the device drivers 506A and 506D and the corresponding secure devices 414A and 414D. The security kernel 504 may prevent unauthorized accesses to the secure device drivers 506A and 506D and the corresponding secure devices 414A and 414D by the application programs 500 and the operating system 502. The device drivers 506B and 506C, on the other hand, may be “non-secure” device drivers, and the corresponding device hardware units 414B and 414C may be “non-secure” device hardware units. The device drivers 506B and 506C and the corresponding device hardware units 414B and 414C may be, for example, “legacy” device drivers and device hardware units.

It is noted that in other embodiments, the security kernel 504 may be part of the operating system 502. In yet other embodiments, the security kernel 504, the device drivers

506A and 506D, and/or the device drivers 506B and 506C may be part of the operating system 502.

Fig. 6 is a diagram of one embodiment of the CPU 402 of the computer system 400 of Fig. 4. In the embodiment of Fig. 6, the CPU 402 includes an execution unit 600, a memory management unit (MMU) 602, a cache unit 604, a bus interface unit (BIU) 606, a set of control registers 608, and a set of secure execution mode (SEM) registers 610. The CPU SCU 416 is located within the MMU 602. As will be described in detail below, the set of SEM registers 610 are used to implement a secure execution mode (SEM) within the computer system 400 of Fig. 4, and the operation of the CPU SCU 416 is governed by the contents of the set of SEM registers 610. SEM registers 610 are accessed (i.e., written to and/or read from) by the security kernel 504 (Fig. 5). The computer system 400 of Fig. 4 may, for example, operate in the SEM when: (i) THE CPU 402 is an x86 processor operating in the x86 protected mode, (ii) memory paging is enabled, and (iii) the contents of SEM registers 610 specify SEM operation.

In general, the contents of the set of control registers 608 govern operation of the CPU 402. Accordingly, the contents of the set of control registers 608 govern operation of the execution unit 600, the MMU 602, the cache unit 604, and/or the BIU 606. The set of control registers 608 may include, for example, the multiple control registers of the x86 processor architecture.

The execution unit 600 of the CPU 402 fetches instructions (e.g., x86 instructions) and data, executes the fetched instructions, and generates signals (e.g., address, data, and control signals) during instruction execution. The execution unit 600 is coupled to the cache unit 604, and may receive instructions from the memory 406 (Fig. 4) via the cache unit 604 and the BIU 606.

The memory 406 (Fig. 4) of the computer system 400 includes multiple memory locations, each having a unique physical address. When operating in protected mode with paging enabled, an address space of the CPU 402 is divided into multiple blocks called page frames or "pages." As described above, only data corresponding to a portion of the pages is stored within the memory 406 at any given time. In the embodiment of Fig. 6, address signals generated by the execution unit 600 during instruction execution represent segmented (i.e., "logical") addresses. As described below, the MMU 602 translates the segmented addresses generated by the execution unit 600 to corresponding physical addresses of the memory 406. The MMU 602 provides the physical addresses to the cache unit 604. The cache unit 604 is a relatively small storage unit used to store instructions and data recently fetched by the execution unit 600. The BIU 606 is coupled between the cache unit 604 and the host bridge 404, and is used to fetch instructions and data not present in the cache unit 604 from the memory 406 via the host bridge 404.

Fig. 7 is a diagram of one embodiment of the MMU 602 of Fig. 6. In the embodiment of Fig. 7, the MMU 602 includes a segmentation unit 700, a paging unit 702, the CPU SCU 416, and selection logic 704 for selecting between outputs of the segmentation unit 700 and the paging unit 702 to produce a physical address. As indicated in Fig. 7, the segmentation unit 700 receives a segmented address from the execution unit 600 and uses a well-known segmented-to-linear address translation mechanism of the x86 processor architecture to produce a corresponding linear address at an output. As indicated in Fig. 7, when enabled by a "PAGING" signal, the paging unit 702 receives the linear addresses produced by the segmentation unit 700 and produces corresponding physical addresses. The PAGING signal may mirror the paging flag (PG) bit in a control register 0 (CR0) of the x86 processor architecture and of the set of control registers 608 (Fig. 6). When the PAGING signal is

deasserted, memory paging is not enabled, and the selection logic 704 produces the linear address received from the segmentation unit 700 as the physical address.

When the PAGING signal is asserted, memory paging is enabled, and the paging unit 702 translates the linear address received from the segmentation unit 700 to a corresponding physical address using the above described linear-to-physical address translation mechanism 100 of the x86 processor architecture (Fig. 1). As described above, during the linear-to-physical address translation operation, the contents of the U/S bits of the selected page directory entry and the selected page table entry are logically ANDed to determine if the access to a page frame is authorized. Similarly, the contents of the R/W bits of the selected page directory entry and the selected page table entry are logically ANDed to determine if the access to the page frame is authorized. If the logical combinations of the U/S and R/W bits indicate the access to the page frame is authorized, the paging unit 702 produces the physical address resulting from the linear-to-physical address translation operation. The selection logic 704 receives the physical address produced by the paging unit 702, produces the physical address received from the paging unit 702 as the physical address, and provides the physical address to the cache unit 604.

On the other hand, if the logical combinations of the U/S and R/W bits indicate the access to the page frame 108 is not authorized, the paging unit 702 does not produce a physical address during the linear-to-physical address translation operation. Instead, the paging unit 702 asserts a general protection fault (GPF) signal, and the MMU 602 forwards the GPF signal to the execution unit 600. In response to the GPF signal, the execution unit 600 may execute an exception handler routine, and may ultimately halt the execution of one of the application programs 500 (Fig. 5) running when the GPF signal was asserted. The paging unit 702 may also include a translation lookaside buffer (TLB) for storing a relatively small number of recently determined linear-to-physical address translations.

Fig. 8 is a diagram of one embodiment of the CPU SCU 416 of Fig. 7. In the embodiment of Fig. 8, the CPU SCU 416 includes security check logic 800 coupled to the set of SEM registers 610 (Fig. 6) and a security attribute table (SAT) entry buffer 802. As described below, SAT entries include additional security information above the U/S and R/W bits of page directory and page table entries corresponding to memory pages. The security check logic 800 uses the additional security information stored within a given SAT entry to prevent unauthorized software-initiated accesses to the corresponding memory page. The SAT entry buffer 802 is used to store a relatively small number of SAT entries of recently accessed memory pages.

As described above, the set of SEM registers 610 are used to implement a secure execution mode (SEM) within the computer system 400 of Fig. 4. The contents of the set of SEM registers 610 govern the operation of the CPU SCU 416. The security check logic 800 receives information to be stored in the SAT entry buffer 802 from the MMU 602 via a communication bus indicated in Fig. 8. The security check logic 800 also receives a physical address produced by the paging unit 702.

Figs. 9-11 will now be used to describe how additional security information of a selected memory page is obtained within the computer system 400 of Fig. 4. Fig. 9 is a diagram of one embodiment of a mechanism 900 for accessing a SAT entry of the selected memory page to obtain additional security information of the selected memory page. The mechanism 900 of Fig. 9 may be embodied within the security check logic 800 of Fig. 8, and may be implemented when the computer system 400 of Fig. 4 is operating in the SEM. The mechanism 900 involves a linear address 902 produced by the segmentation unit 700 (Fig. 7) during execution of an instruction, an SAT directory 904, multiple SATs including a SAT 906, and a SAT base address register 908 of the set of SEM registers 610. The SAT directory 904 and the multiple SATs, including the SAT 906, are SEM data structures created and

maintained by the security kernel 504 (Fig. 5). As described below, the SAT directory 904 (when present) and any needed SAT are copied into the memory 406 before being accessed.

The SAT base address register 908 includes a present (P) bit which indicates the presence of a valid SAT directory base address within the SAT base address register 908. The highest ordered (i.e., most significant) bits of the SAT base address register 908 are reserved for the SAT directory base address. The SAT directory base address is a base address of a memory page containing the SAT directory 904. If P=1, the SAT directory base address is valid, and SAT tables specify the security attributes of memory pages. If P=0, the SAT directory base address is not valid, no SAT tables exist, and security attributes of memory pages are determined by a SAT default register.

Fig. 10 is a diagram of one embodiment of the SAT default register 1000. In the embodiment of Fig. 10, the SAT default register 1000 includes a secure context identification (SCID) field. The SCID field includes multiple bit positions of the SAT default register 1000. The bit positions form a binary representation of an SCID value, wherein the SCID value is an integer value greater than or equal to 0. The SCID value indicates a default security context level for memory pages in the computer system 400. For example, where the SCID field includes  $n$  bits of the SAT default register 1000, a given memory page may occupy one of  $2^n$  possible security context levels in the computer system 400.

The linear address 902 has a corresponding physical address residing in the selected memory page. The corresponding physical address is generated by the paging unit 702 (Fig. 7) using the address translation mechanism 100 of Fig. 1. Referring back to Fig. 9 and assuming the P bit of SAT base address register 908 is a '1', the linear address 902 produced by the segmentation unit 700 (Fig. 7) is divided into three portions to access the SAT entry of the selected memory page. As described above, the SAT directory base address of the SAT base address register 908 is the base address of the memory page containing the SAT



directory 904. The SAT directory 904 includes multiple SAT directory entries, including a SAT directory entry 910. Each SAT directory entry may have a corresponding SAT in the memory 406. An “upper” portion of the linear address 902, including the highest ordered or most significant bits of the linear address 902, is used as an index into the SAT directory 904.

5 The SAT directory entry 910 is selected from within the SAT directory 904 using the SAT directory base address of the SAT base address register 908 and the upper portion of the linear address 902.

Fig. 11 is a diagram of one embodiment of a SAT directory entry format 1100. In accordance with Fig. 11, each SAT directory entry includes a present (P) bit which indicates the presence of a valid SAT base address within the SAT directory entry. In the embodiment of Fig. 11, the highest ordered (i.e., the most significant) bits of each SAT directory entry are reserved for a SAT base address. The SAT base address is a base address of a memory page containing a corresponding SAT. If P=1, the SAT base address is valid, and the corresponding SAT is stored in the memory 406.

10 If P=0, the SAT base address is not valid, and the corresponding SAT does not exist in the memory 406 and must be copied into the memory 406 from a storage device (e.g., a disk drive). If P=0, the security check logic 800 may signal a page fault to logic within the paging unit 702, and the MMU 602 may forward the page fault signal to the execution unit 600 (Fig. 6). In response to the page fault signal, the execution unit 600 may execute a page  
15 fault handler routine which retrieves the needed SAT from the storage device, and store the needed SAT in the memory 406. After the needed SAT is stored in the memory 406, the P bit of the corresponding SAT directory entry is set to ‘1’, and the mechanism 900 is continued. Alternately, if P=0, the contents of a SAT default base address field (not shown) of the SAT default register 1000 (Fig. 10) may be used as the SAT base address.  
20

Referring back to Fig. 9, a “middle” portion of the linear address 902 is used as an index into the SAT 906. The SAT entry 906 is thus selected within the SAT 906 using the SAT base address of the SAT directory entry 910 and the middle portion of the linear address 902. Fig. 12 is a diagram of one embodiment of a SAT entry format 1200. In the  
5 embodiment of Fig. 12, each SAT entry includes a secure context identification (SCID) field. The SCID field includes multiple bit positions of the SAT entry. The bit positions form a binary representation of an SCID value, where the SCID value is an integer value greater than or equal to 0. The SCID value indicates a security context level of the selected memory page. For example, where the SCID field includes  $n$  bits of the SAT entry, the selected memory  
10 page may occupy one of  $2^n$  possible security context levels in the computer system 400.

The BIU 606 (Fig. 6) retrieves needed SEM data structure entries from the memory 406, and provides the SEM data structure entries to the MMU 602. Referring back to Fig. 8, the security check logic 800 receives SEM data structure entries from the MMU 602 and the paging unit 702 via the communication bus. As described above, the SAT entry buffer 802 is  
15 used to store a relatively small number of SAT entries of recently accessed memory pages. The security check logic 800 stores a given SAT entry in the SAT entry buffer 802, along with a “tag” portion of the corresponding physical address.

During a subsequent memory page access, the security check logic 800 may compare a “tag” portion of a physical address produced by the paging unit 702 to tag portions of  
20 physical addresses corresponding to SAT entries stored in the SAT entry buffer 802. If the tag portion of the physical address matches a tag portion of a physical address corresponding to a SAT entry stored in the SAT entry buffer 802, the security check logic 800 may access the SAT entry in the SAT entry buffer 802, eliminating the need to perform the process of Fig. 9 to obtain the SAT entry from the memory 406. The security kernel 504 (Fig. 5)  
25 modifies the contents of the SAT base address register 908 in the CPU 402 (e.g., during

context switches). In response to modifications of the SAT base address register 908, the security check logic 800 of the CPU SCU 416 may flush the SAT entry buffer 802.

In the embodiment of Fig. 7, the CPU SCU 416 is coupled to the paging unit 702. When the computer system 400 of Fig. 4 is operating in the SEM, the security check logic 800 receives the current privilege level (CPL) of the currently executing task (i.e., the currently executing instruction). The security check logic 800 also receives the page directory entry (PDE) U/S bit, the PDE R/W bit, the page table entry (PTE) U/S bit, and the PTE R/W bit of the selected memory page within which the corresponding physical address resides, from the paging unit 702. The security check logic 800 uses the above information, along with the SCID value of the SAT entry corresponding to the selected memory page, to determine if the memory 406 access is authorized.

The CPU 402 of Fig. 6 may be an x86 processor, and may include a code segment (CS) register, one of the 16-bit segment registers of the x86 processor architecture. Each segment register selects a block of memory (e.g., a segment of memory). In the protected mode with paging enabled, the CS register is loaded with a segment selector that indicates an executable block of the memory 406. The highest ordered (i.e., most significant) bits of the segment selector are used to store information indicating a block of memory (e.g., a segment of memory) including a next instruction to be executed by the execution unit 600 of the CPU 402 (Fig. 6). An instruction pointer (IP/EIP/RIP) register is used to store an offset into the block of memory (e.g., the segment of memory) indicated by the CS register. The CS:IP/EIP/RIP pair indicate a segmented address of the next instruction. The two lowest ordered (i.e., least significant) bits of the CS register are used to store a value indicating a current privilege level (CPL) of a task currently being executed by the execution unit 600 (i.e., the CPL of the current task).

Table 1 below illustrates exemplary rules for CPU-initiated (i.e., software-initiated) memory accesses when the computer system 400 of Fig. 4 is operating in the SEM. The CPU SCU 416 (Figs. 4-8) and the security kernel 504 (Fig. 5) work together to implement the rules of Table 1 when the computer system 400 of Fig. 4 is operating in the SEM to provide additional security for data stored in the memory 406 over and above data security provided by the operating system 502 (Fig. 5).

In Table 1 below, "CEI-SCID" represents the SCID value of the currently executing instruction, and "SMP-SCID" represents the SCID value of the selected memory page. The CEI-SCID value is the SCID value of the SAT entry corresponding to the memory page containing the currently executing instruction. In the embodiment of Table 1, lower SCID values are associated with higher security context levels. In other embodiments, higher SCID values may be associated with higher security context levels.

Table 1. Exemplary Rules For Software-Initiated Memory Accesses  
When The Computer System 400 Of Fig. 4 Is Operating In The SEM.

Selected Memory Page		Permitted		Remarks
SCID Comparison	CPL U/S R/W	Access		
CEI-SCID > SMP-SCID	X X X	None		SEM Security Exception asserted.
CEI-SCID <= SMP-SCID	0 X 1(R/W)	R/W		Normal x86 protections apply: Full access granted.

CEI-SCID <= SMP-SCID	0	X	0(R)	Read only	Normal x86 protections apply: Only read access.
CEI-SCID <= SMP-SCID	3	0	X	None	Normal x86 protections apply: GPF asserted.
CEI-SCID <= SMP-SCID	3	1	1(R/W)	R/W	Normal x86 protections apply: Full access granted.
CEI-SCID <= SMP-SCID	3	1	0(R)	Read only	Normal x86 protections apply: Only read access.

In Table 1 above, the “CEI-SCID > SMP-SCID” condition is true when the integer value represented by the SCID value of the currently executing instruction is numerically greater than the integer value represented by the SCID value of the selected memory page. The “CEI-SCID <= SMP-SCID” condition is true when the integer value represented by the SCID value of the currently executing instruction is numerically less than or equal to the integer value represented by the SCID value of the selected memory page. The U/S bit of the selected memory page is the logical AND of the PDE U/S bit and the PTE U/S bit of the selected memory page. The R/W bit of the selected memory page is the logical AND of the PDE R/W bit and the PTE R/W bit of the selected memory page. The symbol “X” signifies a “don’t care”: the logical value may be either a ‘0’ or a ‘1’.

Referring back to Fig. 8, the security check logic 800 of the CPU SCU 416 produces a general protection fault (“GPF”) signal and a “SEM SECURITY EXCEPTION” signal, and

provides the GPF and the SEM SECURITY EXCEPTION signals to logic within the MMU 602. When the security check logic 800 asserts the GPF signal, the MMU 602 forwards the GPF signal to the execution unit 600 (Fig. 6). In response to the GPF signal, the execution unit 600 may use the well-known interrupt descriptor table (IDT) vectoring mechanism of the x86 processor architecture to access and execute a GPF handler routine.

When the security check logic 800 asserts the SEM SECURITY EXCEPTION signal, the MMU 602 forwards the SEM SECURITY EXCEPTION signal to the execution unit 600. Unlike normal processor exceptions which use the IDT vectoring mechanism of the x86 processor architecture, a different vectoring method may be used to handle SEM security exceptions. SEM security exceptions may be dispatched through a pair of registers (e.g., model specific registers or MSRs) similar to the way x86 "SYSENTER" and "SYSEXIT" instructions operate. The pair of registers may be "security exception entry point" registers, and may define a branch target address for instruction execution when a SEM security exception occurs. The security exception entry point registers may define the code segment (CS), then instruction pointer (IP, or the 64-bit version RIP), stack segment (SS), and the stack pointer (SP, or the 64-bit version RSP) values to be used on entry to a SEM security exception handler. Under software control, the execution unit 600 (Fig. 6) may push the previous SS, SP/RSP, EFLAGS, CS, and IP/RIP values onto a new stack to indicate where the exception occurred. In addition, the execution unit 600 may push an error code onto the stack. It is noted that a normal return from interrupt (IRET) instruction may not be used as the previous SS and SP/RSP values are always saved, and a stack switch is always accomplished, even if a change in CPL does not occur. Accordingly, a new instruction may be defined to accomplish a return from the SEM security exception handler.

Figs. 13A and 13B in combination form a flow chart of one embodiment of a method 1300 for providing access security for a memory used to store data arranged within multiple

memory pages. The method 1300 may, for example, reflect the exemplary rules of Table 1 for CPU-initiated (i.e., software-initiated) memory accesses when the computer system 400 of Fig. 4 is operating in the SEM. The method 1300 may be embodied within the MMU 602 (Figs. 6-7).

5 During a step 1302 of the method 1300, a linear address produced during execution of an instruction is received, along with a security attribute of the instruction (e.g., a CPL of a task including the instruction). The instruction resides in a first memory page. During a step 1304, the linear address is used to access at least one paged memory data structure located in the memory (e.g., a page directory and a page table) to obtain a base address of a selected memory page and security attributes of the selected memory page. The security attributes of the selected memory page may include, for example, a U/S bit and a R/W bit of a page directory entry and a U/S bit and a R/W bit of a page table entry.

During a decision step 1306, the security attribute of the instruction and the security attributes of the selected memory page are used to determine whether or not the access is authorized. If the access is authorized, the base address of the selected memory page and an offset are combined during a step 1308 to produce a physical address within the selected memory page. If the access is not authorized, a fault signal (e.g., a general protection fault signal or GPF signal) is generated during a step 1310.

During a step 1312, which may be carried out substantially at the same time as the step 1304 (i.e., substantially in parallel with the step 1304), at least one security attribute data structure located in the memory (e.g., the SAT directory 904 of Fig. 9 and a SAT) is accessed using the linear address produced during execution of the instruction to obtain an additional security attribute of the first memory page, wherein the first memory page includes the instruction, and an additional security attribute of the selected memory page, wherein the selected memory page includes the physical address corresponding to the linear address. The

additional security attribute of the first memory page may include, for example, a security context identification (SCID) value as described above, wherein the SCID value indicates a security context level of the first memory page. Similarly, the additional security attribute of the selected memory page may include an SCID value, wherein the SCID value indicates a security context level of the selected memory page.

During a step 1314, a numerical value conveyed by the additional security attribute of the first memory page is compared to a numerical value conveyed by the additional security attribute of selected memory page. The selected memory page is accessed during a step 1316 dependent upon a result of the comparing of the numerical values conveyed by the security attribute of the first memory page and the additional security attribute of selected memory page.

For example, where the method 1300 reflects the exemplary rules of Table 1 above for CPU-initiated (i.e., software-initiated) memory accesses when the computer system 400 of Fig. 4 is operating in the SEM, the selected memory page may be accessed during the step 1316 only if the integer value represented by the SCID value of the instruction is numerically less than or equal to the integer value represented by the SCID value of the selected memory page (i.e., if  $CEI\_SCID \leq SMP\_SCID$ ). If, on the other hand, the integer value represented by the SCID value of the instruction is numerically greater than the integer value represented by the SCID value of the selected memory page (i.e., if  $CEI\_SCID > SMP\_SCID$ ), the selected memory page may not be accessed during the step 1316. In this situation, the SEM SECURITY EXCEPTION signal may be asserted as indicated in Table 1 and described above.

The particular embodiments disclosed above are illustrative only, as the invention may be modified and practiced in different but equivalent manners apparent to those skilled in the art having the benefit of the teachings herein. Furthermore, no limitations are intended



to the details of construction or design herein shown, other than as described in the claims below. It is therefore evident that the particular embodiments disclosed above may be altered or modified and all such variations are considered within the scope and spirit of the invention. Accordingly, the protection sought herein is as set forth in the claims below.

1  
2  
3  
4  
5  
6  
7  
8  
9  
10  
11  
12  
13  
14  
15  
16  
17  
18  
19  
20  
21  
22  
23  
24  
25  
26  
27  
28  
29  
30  
31  
32  
33  
34  
35  
36  
37  
38  
39  
40  
41  
42  
43  
44  
45  
46  
47  
48  
49  
50  
51  
52  
53  
54  
55  
56  
57  
58  
59  
60  
61  
62  
63  
64  
65  
66  
67  
68  
69  
70  
71  
72  
73  
74  
75  
76  
77  
78  
79  
80  
81  
82  
83  
84  
85  
86  
87  
88  
89  
90  
91  
92  
93  
94  
95  
96  
97  
98  
99  
100  
101  
102  
103  
104  
105  
106  
107  
108  
109  
110  
111  
112  
113  
114  
115  
116  
117  
118  
119  
120  
121  
122  
123  
124  
125  
126  
127  
128  
129  
130  
131  
132  
133  
134  
135  
136  
137  
138  
139  
140  
141  
142  
143  
144  
145  
146  
147  
148  
149  
150  
151  
152  
153  
154  
155  
156  
157  
158  
159  
160  
161  
162  
163  
164  
165  
166  
167  
168  
169  
170  
171  
172  
173  
174  
175  
176  
177  
178  
179  
180  
181  
182  
183  
184  
185  
186  
187  
188  
189  
190  
191  
192  
193  
194  
195  
196  
197  
198  
199  
200  
201  
202  
203  
204  
205  
206  
207  
208  
209  
210  
211  
212  
213  
214  
215  
216  
217  
218  
219  
220  
221  
222  
223  
224  
225  
226  
227  
228  
229  
230  
231  
232  
233  
234  
235  
236  
237  
238  
239  
240  
241  
242  
243  
244  
245  
246  
247  
248  
249  
250  
251  
252  
253  
254  
255  
256  
257  
258  
259  
260  
261  
262  
263  
264  
265  
266  
267  
268  
269  
270  
271  
272  
273  
274  
275  
276  
277  
278  
279  
280  
281  
282  
283  
284  
285  
286  
287  
288  
289  
290  
291  
292  
293  
294  
295  
296  
297  
298  
299  
300  
301  
302  
303  
304  
305  
306  
307  
308  
309  
310  
311  
312  
313  
314  
315  
316  
317  
318  
319  
320  
321  
322  
323  
324  
325  
326  
327  
328  
329  
330  
331  
332  
333  
334  
335  
336  
337  
338  
339  
340  
341  
342  
343  
344  
345  
346  
347  
348  
349  
350  
351  
352  
353  
354  
355  
356  
357  
358  
359  
360  
361  
362  
363  
364  
365  
366  
367  
368  
369  
370  
371  
372  
373  
374  
375  
376  
377  
378  
379  
380  
381  
382  
383  
384  
385  
386  
387  
388  
389  
390  
391  
392  
393  
394  
395  
396  
397  
398  
399  
400  
401  
402  
403  
404  
405  
406  
407  
408  
409  
410  
411  
412  
413  
414  
415  
416  
417  
418  
419  
420  
421  
422  
423  
424  
425  
426  
427  
428  
429  
430  
431  
432  
433  
434  
435  
436  
437  
438  
439  
440  
441  
442  
443  
444  
445  
446  
447  
448  
449  
450  
451  
452  
453  
454  
455  
456  
457  
458  
459  
460  
461  
462  
463  
464  
465  
466  
467  
468  
469  
470  
471  
472  
473  
474  
475  
476  
477  
478  
479  
480  
481  
482  
483  
484  
485  
486  
487  
488  
489  
490  
491  
492  
493  
494  
495  
496  
497  
498  
499  
500  
501  
502  
503  
504  
505  
506  
507  
508  
509  
510  
511  
512  
513  
514  
515  
516  
517  
518  
519  
520  
521  
522  
523  
524  
525  
526  
527  
528  
529  
530  
531  
532  
533  
534  
535  
536  
537  
538  
539  
540  
541  
542  
543  
544  
545  
546  
547  
548  
549  
550  
551  
552  
553  
554  
555  
556  
557  
558  
559  
560  
561  
562  
563  
564  
565  
566  
567  
568  
569  
570  
571  
572  
573  
574  
575  
576  
577  
578  
579  
580  
581  
582  
583  
584  
585  
586  
587  
588  
589  
590  
591  
592  
593  
594  
595  
596  
597  
598  
599  
600  
601  
602  
603  
604  
605  
606  
607  
608  
609  
610  
611  
612  
613  
614  
615  
616  
617  
618  
619  
620  
621  
622  
623  
624  
625  
626  
627  
628  
629  
630  
631  
632  
633  
634  
635  
636  
637  
638  
639  
640  
641  
642  
643  
644  
645  
646  
647  
648  
649  
650  
651  
652  
653  
654  
655  
656  
657  
658  
659  
660  
661  
662  
663  
664  
665  
666  
667  
668  
669  
670  
671  
672  
673  
674  
675  
676  
677  
678  
679  
680  
681  
682  
683  
684  
685  
686  
687  
688  
689  
690  
691  
692  
693  
694  
695  
696  
697  
698  
699  
700  
701  
702  
703  
704  
705  
706  
707  
708  
709  
710  
711  
712  
713  
714  
715  
716  
717  
718  
719  
720  
721  
722  
723  
724  
725  
726  
727  
728  
729  
730  
731  
732  
733  
734  
735  
736  
737  
738  
739  
740  
741  
742  
743  
744  
745  
746  
747  
748  
749  
750  
751  
752  
753  
754  
755  
756  
757  
758  
759  
760  
761  
762  
763  
764  
765  
766  
767  
768  
769  
770  
771  
772  
773  
774  
775  
776  
777  
778  
779  
780  
781  
782  
783  
784  
785  
786  
787  
788  
789  
790  
791  
792  
793  
794  
795  
796  
797  
798  
799  
800  
801  
802  
803  
804  
805  
806  
807  
808  
809  
810  
811  
812  
813  
814  
815  
816  
817  
818  
819  
820  
821  
822  
823  
824  
825  
826  
827  
828  
829  
830  
831  
832  
833  
834  
835  
836  
837  
838  
839  
840  
841  
842  
843  
844  
845  
846  
847  
848  
849  
850  
851  
852  
853  
854  
855  
856  
857  
858  
859  
860  
861  
862  
863  
864  
865  
866  
867  
868  
869  
870  
871  
872  
873  
874  
875  
876  
877  
878  
879  
880  
881  
882  
883  
884  
885  
886  
887  
888  
889  
890  
891  
892  
893  
894  
895  
896  
897  
898  
899  
900  
901  
902  
903  
904  
905  
906  
907  
908  
909  
910  
911  
912  
913  
914  
915  
916  
917  
918  
919  
920  
921  
922  
923  
924  
925  
926  
927  
928  
929  
930  
931  
932  
933  
934  
935  
936  
937  
938  
939  
940  
941  
942  
943  
944  
945  
946  
947  
948  
949  
950  
951  
952  
953  
954  
955  
956  
957  
958  
959  
960  
961  
962  
963  
964  
965  
966  
967  
968  
969  
970  
971  
972  
973  
974  
975  
976  
977  
978  
979  
980  
981  
982  
983  
984  
985  
986  
987  
988  
989  
990  
991  
992  
993  
994  
995  
996  
997  
998  
999  
1000  
1001  
1002  
1003  
1004  
1005  
1006  
1007  
1008  
1009  
1010  
1011  
1012  
1013  
1014  
1015  
1016  
1017  
1018  
1019  
1020  
1021  
1022  
1023  
1024  
1025  
1026  
1027  
1028  
1029  
1030  
1031  
1032  
1033  
1034  
1035  
1036  
1037  
1038  
1039  
1040  
1041  
1042  
1043  
1044  
1045  
1046  
1047  
1048  
1049  
1050  
1051  
1052  
1053  
1054  
1055  
1056  
1057  
1058  
1059  
1060  
1061  
1062  
1063  
1064  
1065  
1066  
1067  
1068  
1069  
1070  
1071  
1072  
1073  
1074  
1075  
1076  
1077  
1078  
1079  
1080  
1081  
1082  
1083  
1084  
1085  
1086  
1087  
1088  
1089  
1090  
1091  
1092  
1093  
1094  
1095  
1096  
1097  
1098  
1099  
1100  
1101  
1102  
1103  
1104  
1105  
1106  
1107  
1108  
1109  
1110  
1111  
1112  
1113  
1114  
1115  
1116  
1117  
1118  
1119  
1120  
1121  
1122  
1123  
1124  
1125  
1126  
1127  
1128  
1129  
1130  
1131  
1132  
1133  
1134  
1135  
1136  
1137  
1138  
1139  
1140  
1141  
1142  
1143  
1144  
1145  
1146  
1147  
1148  
1149  
1150  
1151  
1152  
1153  
1154  
1155  
1156  
1157  
1158  
1159  
1160  
1161  
1162  
1163  
1164  
1165  
1166  
1167  
1168  
1169  
1170  
1171  
1172  
1173  
1174  
1175  
1176  
1177  
1178  
1179  
1180  
1181  
1182  
1183  
1184  
1185  
1186  
1187  
1188  
1189  
1190  
1191  
1192  
1193  
1194  
1195  
1196  
1197  
1198  
1199  
1200  
1201  
1202  
1203  
1204  
1205  
1206  
1207  
1208  
1209  
1210  
1211  
1212  
1213  
1214  
1215  
1216  
1217  
1218  
1219  
1220  
1221  
1222  
1223  
1224  
1225  
1226  
1227  
1228  
1229  
1230  
1231  
1232  
1233  
1234  
1235  
1236  
1237  
1238  
1239  
1240  
1241  
1242  
1243  
1244  
1245  
1246  
1247  
1248  
1249  
1250  
1251  
1252  
1253  
1254  
1255  
1256  
1257  
1258  
1259  
1260  
1261  
1262  
1263  
1264  
1265  
1266  
1267  
1268  
1269  
1270  
1271  
1272  
1273  
1274  
1275  
1276  
1277  
1278  
1279  
1280  
1281  
1282  
1283  
1284  
1285  
1286  
1287  
1288  
1289  
1290  
1291  
1292  
1293  
1294  
1295  
1296  
1297  
1298  
1299  
1300  
1301  
1302  
1303  
1304  
1305  
1306  
1307  
1308  
1309  
1310  
1311  
1312  
1313  
1314  
1315  
1316  
1317  
1318  
1319  
1320  
1321  
1322  
1323  
1324  
1325  
1326  
1327  
1328  
1329  
1330  
1331  
1332  
1333  
1334  
1335  
1336  
1337  
1338  
1339  
1340  
1341  
1342  
1343  
1344  
1345  
1346  
1347  
1348  
1349  
1350  
1351  
1352  
1353  
1354  
1355  
1356  
1357  
1358  
1359  
1360  
1361  
1362  
1363  
1364  
1365  
1366  
1367  
1368  
1369  
1370  
1371  
1372  
1373  
1374  
1375  
1376  
1377  
1378  
1379  
1380  
1381  
1382  
1383  
1384  
1385  
1386  
1387  
1388  
1389  
1390  
1391  
1392  
1393  
1394  
1395  
1396  
1397  
1398  
1399  
1400  
1401  
1402  
1403  
1404  
1405  
1406  
1407  
1408  
1409  
1410  
1411  
1412  
1413  
1414  
1415  
1416  
1417  
1418  
1419  
1420  
1421  
1422  
1423  
1424  
1425  
1426  
1427  
1428  
1429  
1430  
1431  
1432  
1433  
1434  
1435  
1436  
1437  
1438  
1439  
1440  
1441  
1442  
1443  
1444  
1445  
1446  
1447  
1448  
1449  
1450  
1451  
1452  
1453  
1454  
1455  
1456  
1457  
1458  
1459  
1460  
1461  
1462  
1463  
1464  
1465  
1466  
1467  
1468  
1469  
1470  
1471  
1472  
1473  
1474  
1475  
1476  
1477  
1478  
1479  
1480  
1481  
1482  
1483  
1484  
1485  
1486  
1487  
1488  
1489  
1490  
1491  
1492  
1493  
1494  
1495  
1496  
1497  
1498  
1499  
1500  
1501  
1502  
1503  
1504  
1505  
1506  
1507  
1508  
1509  
1510  
1511  
1512  
1513  
1514  
1515  
1516  
1517  
1518  
1519  
1520  
1521  
1522  
1523  
1524  
1525  
1526  
1527  
1528  
1529  
1530  
1531  
1532  
1533  
1534  
1535  
1536  
1537  
1538  
1539  
1540  
1541  
1542  
1543  
1544  
1545  
1546  
1547  
1548  
1549  
1550  
1551  
1552  
1553  
1554  
1555  
1556  
1557  
1558  
1559  
1560  
1561  
1562  
1563  
1564  
1565  
1566  
1567  
1568  
1569  
1570  
1571  
1572  
1573  
1574  
1575  
1576  
1577  
1578  
1579  
1580  
1581  
1582  
1583  
1584  
1585  
1586  
1587  
1588  
1589  
1590  
1591  
1592  
1593  
1594  
1595  
1596  
1597  
1598  
1599  
1600  
1601  
1602  
1603  
1604  
1605  
1606  
1607  
1608  
1609  
1610  
1611  
1612  
1613  
1614  
1615  
1616  
1617  
1618  
1619  
1620  
1621  
1622  
1623  
1624  
1625  
1626  
1627  
1628  
1629  
1630  
1631  
1632  
1633  
1634  
1635  
1636  
1637  
1638  
1639  
1640  
1641  
1642  
1643  
1644  
1645  
1646  
1647  
1648  
1649  
1650  
1651  
1652  
1653  
1654  
1655  
1656  
1657  
1658  
1659  
1660  
1661  
1662  
1663  
1664  
1665  
1666  
1667  
1668  
1669  
1670  
1671  
1672  
1673  
1674  
1675  
1676  
1677  
1678  
1679  
1680  
1681  
1682  
1683  
1684  
1685  
1686  
1687  
1688  
1689  
1690  
1691  
1692  
1693  
1694  
1695  
1696  
1697  
1698  
1699  
1700  
1701  
1702  
1703  
1704  
1705  
1706  
1707  
1708  
1709  
1710  
1711  
1712  
1713  
1714  
1715  
1716  
1717  
1718  
1719  
1720  
1721  
1722  
1723  
1724  
1725  
1726  
1727  
1728  
1729  
1730  
1731  
1732  
1733  
1734  
1735  
1736  
1737  
1738  
1739  
1740  
1741  
1742  
1743  
1744  
1745  
1746  
1747  
1748  
1749  
1750  
1751  
1752  
1753  
1754  
1755  
1756  
1757  
1758  
1759  
1760  
1761  
1762  
1763  
1764  
1765  
1766  
1767  
1768  
1769  
1770  
1771  
1772  
1773  
1774  
1775  
1776  
1777  
1778  
1779  
1780  
1781  
1782  
1783  
1784  
1785  
1786  
1787  
1788  
1789  
1790  
1791  
1792  
1793  
1794  
1795  
1796  
1797  
1798  
1799  
1800  
1801  
1802  
1803  
1804  
1805  
1806  
1807  
1808  
1809  
1810  
1811  
1812  
1813  
1814  
1815  
1816  
1817  
1818  
1819  
1820  
1821  
1822  
1823  
1824  
1825  
1826  
1827  
1828  
1829  
1830  
1831  
1832  
1833  
1834  
1835  
1836  
1837  
1838  
1839  
1840  
1841  
1842  
1843  
1844  
1845  
1846  
1847  
1848  
1849  
1850  
1851  
1852  
1853  
1854  
1855  
1856  
1857  
1858  
1859  
1860  
1861  
1862  
1863  
1864  
1865  
1866  
1867  
1868  
1869  
1870  
1871  
1872  
1873  
1874  
1875  
1876  
1877  
1878  
1879  
1880  
1881  
1882  
1883  
1884  
1885  
1886  
1887  
1888  
1889  
1890  
1891  
1892  
1893  
1894  
1895  
1896  
1897  
1898  
1899  
1900  
1901  
1902  
1903  
1904  
1905  
1906  
1907  
1908  
1909  
1910  
1911  
1912  
1913  
1914  
1915  
1916  
1917  
1918  
1919  
1920  
1921  
1922  
1923  
1924  
1925  
1926  
1927  
1928  
1929  
1930  
1931  
1932  
1933  
1934  
1935  
1936  
1937  
1938  
1939  
1940  
1941  
1942  
1943  
1944  
1945  
1946  
1947  
1948  
1949  
1950  
1951  
1952  
1953  
1954  
1955  
1956  
1957  
1958  
1959  
1960  
1961  
1962  
1963  
1964  
1965  
1966  
1967  
1968  
1969  
1970  
1971  
1972  
1973  
1974  
1975  
1976  
1977  
1978  
1979  
1980  
1981  
1982  
1983  
1984  
1985  
1986  
1987  
1988  
1989  
1990  
1991  
1992  
1993  
1994  
1995  
1996  
1997  
1998  
1999  
2000  
2001  
2002  
2003  
2004  
2005  
2006  
2007  
2008  
2009  
2010  
2011  
2012  
2013  
2014  
2015  
2016  
2017  
2018  
2019  
2020  
2021  
2022  
2023  
2024  
2025  
2026  
2027  
2028  
2029  
2030  
2031  
2032  
2033  
2034  
2035  
2036  
2037  
2038  
2039  
2040  
2041  
2042  
2043  
2044  
2045  
2046  
2047  
2048  
2049  
2050  
2051  
2052  
2053  
2054  
2055  
2056  
2057  
2058  
2059  
2060  
2061  
2062  
2063  
2064  
2065  
2066  
2067  
2068  
2069  
2070  
2071  
2072  
2073  
2074  
2075  
2076  
2077  
2078  
2079  
2080  
2081  
2082  
2083  
2084  
2085  
2086  
2087  
2088  
2089  
2090  
2091  
2092  
2093  
2094  
2095  
2096  
2097  
2098  
2099  
2100  
2101  
2102  
2103  
2104  
2105  
2106  
2107  
2108  
2109  
2110  
2111  
2112  
2113  
2114  
2115  
2116  
2117  
2118  
2119  
2120  
2121  
2122  
2123  
2124  
2125  
2126  
2127  
2128  
2129  
2130  
2131  
2132  
2133  
2134  
2135  
2136  
2137  
2138  
2139  
2140  
2141  
2142  
2143  
2144  
2145  
2146  
2147  
2148  
2149  
2150  
2151  
2152  
2153  
2154  
2155  
2156  
2157  
2158  
2159  
2160  
2161  
2162  
2163  
2164  
2165  
2166  
2167  
2168  
2169  
2170  
2171  
2172  
2173  
2174  
2175  
2176  
2177  
2178  
2179  
2180  
2181  
2182  
2183  
2184  
2185  
2186  
2187  
2188  
2189  
2190  
2191  
2192  
2193  
2194  
2195  
2196  
2197  
2198  
2199  
2200  
2201  
2202  
2203  
2204  
2205  
2206  
2207  
2208  
2209  
2210  
2211  
2212  
2213  
2214  
22